TED (10)-4070 **Reg. No. ……………………**

o(REVISION—2010) **Signature…………………...**

FOURTH SEMESTER DIPLOMA EXAMINATION IN ENGINEERING/
TECHNOLOGY—MARCH, 2014

DATABASE MANAGEMENT SYSTEMS

(Common for CT and IF)

[*Time* : 3 hours

(Maximum marks : 100)

PART—A

I. Answer the following questions in one or two sentences. Each question carries 2 marks.

1. **Define Database.**

Database: A collection of related data.2.

2. **Define Tuple in database.**

A tuple is an ordered set of values

Each value is derived from an appropriate domain

3. **Write the different types of join operations in relational algebra.**
   - Natural Join
   - Projection Join
   - Equi Join
4. **Define Un-Normalized Relation.**

   In Un normalized relation data can repeat within a column
5. **Write the SQL command to define the structure of a relation and to add tuples in that relation.**

   a) **define the structure of a relation :**

   SQL>Desc Table Name

   b) **Add tuples in the relation**:

   SQL>alter table Table name add (Tuple name);

PART B

II. Answer *any five* of the following. Each question carries 6 marks.

1. **Define database model. Describe the various classifications of database models.**

■ **Hierarchical Database Model**

- A segment represents one or a few fields
- Segment's type determines the list and types of fields in the segment
- Segment instance – particular data corresponding to the segment type
- A parent-child relationship – 1:N relationship between two segment types

■ **Network Database Model**

o Two data structures: records and sets

o Records:

- are classified into record types
- Two record types participate in owner-member 1:N relationships

o Each record type is assigned:

- Type field which distinguishes each record type
- Pointer field for each set type in which it participates as an owner or a member

■ **Relational Data Model**

o Commercial products appeared in the late 70s - earlier 80s

o SQL – standard data management language for relational DBMSs

o Data manipulation is supported by theory of sets, relational algebra and calculus

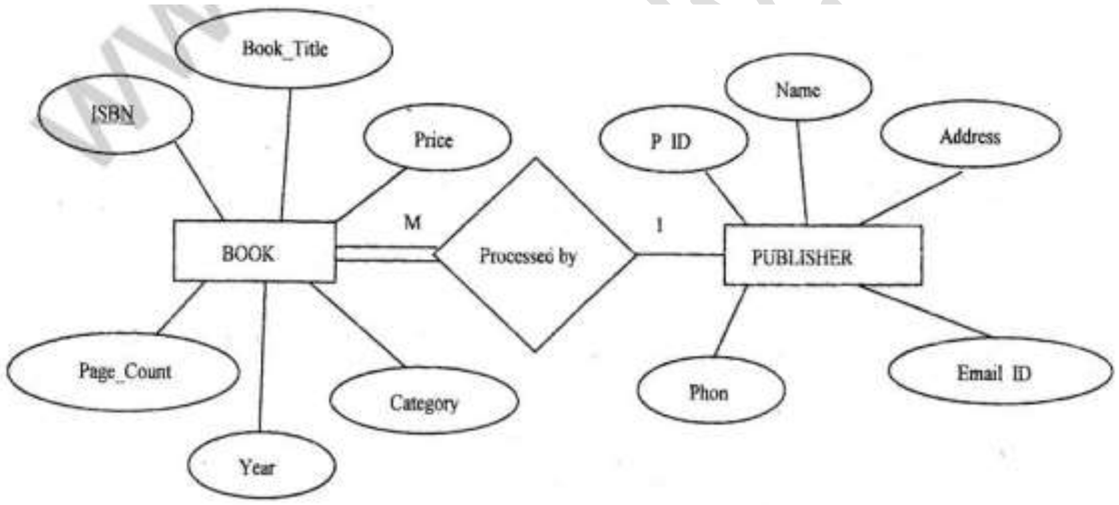
2. **Distinguish the functions of :**
   (i) Query processor

■ Query processor takes care of arranging the underlying access routines to satisfy a given query Thus queries can be specified in terms of the required results rather than in terms of how to achieve those results

(ii) Query optimizer

■ Query optimization is a function of many relational database management systems in which multiple query plans for satisfying a query are examined and a good query plan is identified. This may or not be the absolute best strategy because there are many waysof doing plans. There is a trade-off between the amount of time spent figuring out the best plan and the amount running the plan.

3. Identify the relationship types and entity types in the following ER diagram :



- The Relationship between BOOK and PUBLISHER is Many to One Relation(M:N).
- The ER diagram contains Two entities named BOOK and PUBLISHER. These two entities are strong entities

4. Differentiate between primary key , candidate key and super key.

- A *super key* of an entity set is a set of one or more attributes whose values uniquely determine each entity.
- A *candidate key* of an entity set is a minimal super key
- Although several candidate keys may exist, one of the candidate keys is selected to be the *primary key*.

5. Define constraint. Explain the various constraints in relational database.

Constraints are *conditions* that must hold on *all* valid relation instances. There are three main types of constraints:

- Key constraints
- Entity integrity constraints
- Referential integrity constraints

  Key constraints: If a relation has *several* **candidate keys**, one is chosen arbitrarily to be the **primary key**. The primary key attributes are *underlined.*

- **Entity Integrity**: The *primary key attributes* PK of each relation schema R in S cannot have null values in any tuple of r(R). This is because primary key values are used to *identify* the individual tuples.

  $t[PK] \neq$ null for any tuple t in r(R)

- Note: Other attributes of R may be similarly constrained to disallow null values, even though they are not members of the primary key.

Referential Integrity:

- A constraint involving *two* relations (the previous constraints involve a *single* relation).
- Used to specify a *relationship* among tuples in two relations: the **referencing relation** and the **referenced relation**.
- Tuples in the *referencing relation* R1 have attributes FK (called **foreign key** attributes) that reference the primary key attributes PK of the *referenced relation* R_2. A tuple t_1 in R_1 is said to **reference** a tuple t_2 in R_2 if $t_1[FK] = t_2[PK]$.
- A referential integrity constraint can be displayed in a relational database schema as a directed arc from R_1.FK to R_2.

6. Explain functional dependency with example.

Functional Dependencies

Definition:

Two tuples that agree on the attributes $A_1,\ldots,A_n$

must also agree on the attributes $B_1,\ldots, B_m$

Formally: $A , A , \ldots A \longrightarrow B , B , \ldots B$

If $A_1,\ldots,A_n$ is a key, then

$A_1,\ldots,A_n$ Attributes(R) - $A_1,\ldots,A_n$

Name	SSN	Phone Number
Fred	123-321-99	(201) 555-1234
Fred	123-321-99	(206) 572-4312
Joe	909-438-44	(908) 464-0028
Joe	909-438-44	(212) 555-4000

Problems:

- redundancy
- update anomalies
- deletion anomalies

7. Explain the use of dynamic SQL with an example.

The Dynamic SQL is used to write programs that mention SQL statements whose full text is not known until runtime. The complete query or the procedure is evaluated only at the run time, which gives results depending on the code. In this code some the code may refer the code which is precompiled or may refer some code which is not part of current code. Instead the static SQL statements do not change from execution to execution. The full code of static SQL statements is known at compilation, which gives the following benefits:

 The total compilation checks that the necessary privileges are already given to access the database objects.

 The total compilation verifies that the SQL statements reference valid database objects.

```
CREATE OR REPLACE PROCEDURE my_emp_handler_1(e number) AS BEGIN
        -- Code to process the event
        RETURN;
        END;
```

/

PART—C

(Answer one full question from each unit. Each question carries 15 marks.)

UNIT--I

III. (a) Explain the advantages and disadvantages of DBMS.

- **Advantages**
  - Controlling redundancy in data storage and in development and maintenence efforts.
  - Sharing of data among multiple users.
  - Restricting unauthorized access to data.
    - Providing persistent storage for program Objects (in Object-oriented DBMS's – Providing Storage Structures for efficient Query Processing
- **Disadvantages**
  - A DBMS is a complex software
  - Requires the large amount of primary memory and secondary storage
  - High cost
  - The centralization of all data resources increases vulnerability of the system

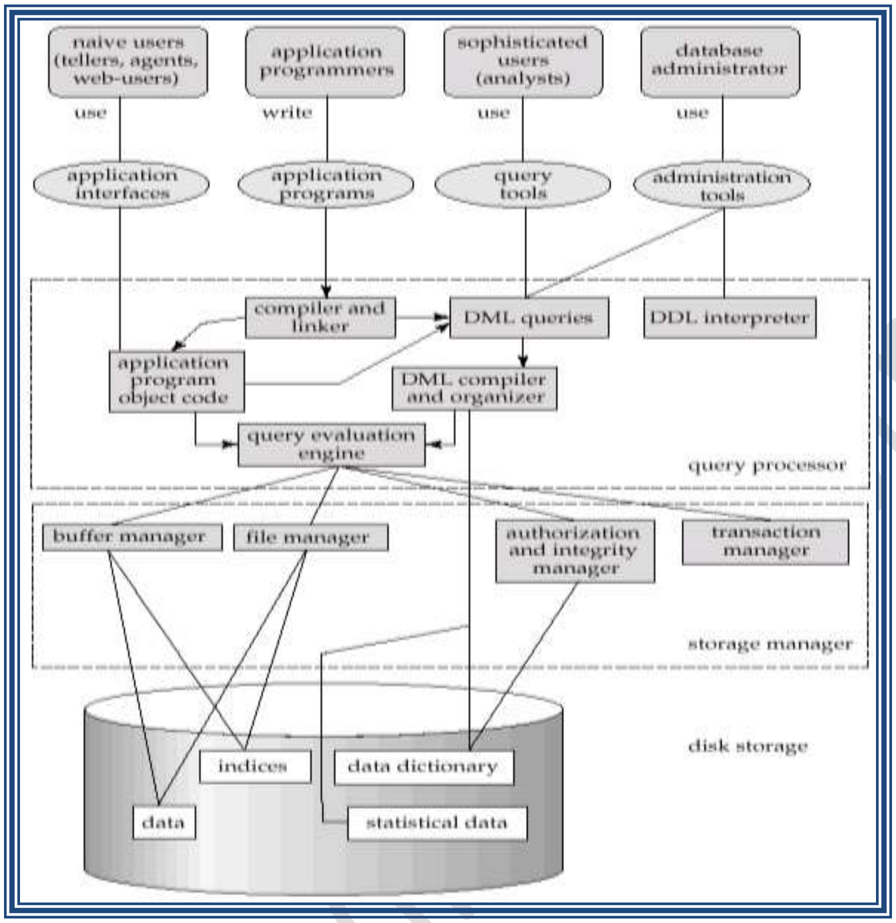
(b) Describe DBMS architecture with components models.

Storage Management

- Storage manager is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible to the following tasks:
  - interaction with the file manager
  - efficient storing, retrieving and updating of data

Concurrency Control

- Concurrent execution of user programs is essential for good DBMS performance.
  - Because disk accesses are frequent, and relatively slow, it is important to keep the cpu humming by working on several user programs concurrently.



Transaction Management

- A *transaction* is a collection of operations that performs a single logical function in a database application
- Transaction-management component ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.
- Concurrency-control manager controls the interaction among the concurrent transactions, to ensure the consistency of the database.

OR

IV (a) Compare the various database users with examples.

- **Database administrators:** responsible for authorizing access to the database, for co-ordinating and monitoring its use, acquiring software, and hardware resources, controlling its use and monitoring efficiency of operations.
- **Database Designers:** responsible to define the content, the structure, the constraints, and functions or transactions against the database. They must communicate with the end-users and understand their needs.
- **End-users**: they use the data for queries, reports and some of them actually update the database content
- **Casual :** access database occasionally when needed
- **Naïve or Parametric :** they make up a large section of the end-user population. They use previously well-defined functions in the form of "canned transactions" against the database. Examples are bank-tellers or reservation clerks who do this activity for an entire shift of operations.
- **Sophisticated** : these include business analysts, scientists, engineers, others thoroughly familiar with the system capabilities. Many use tools in the form of software packages that work closely with the stored database.
- **Stand-alone** : mostly maintain personal databases using ready-to-use packaged applications. An example is a tax program user that creates his or her own internal database.

(b) Define data independence. Distinguish between physical data independence and logical data independence.

- **Physical Data Independence –** the ability to modify the physical schema without changing the application programs
  - Applications depend on the logical schema
  - DBA may change physical level (tuning) without affecting applications
  - The DBMS automatically make the required adjustments, and application programs are not changed (queries may need to be recompiled and optimized…)
- **Logical Data Independence –** the ability to modify the logical schema without changing the application programs
  - Applications depend on the logical schema via the Views
  - Can be supported on a limited basis only (if view is not affected)

UNIT II

V. **(a). Define strong entity.**

- Some entity sets in real world naturally independ from other set
- They can be uniquely identified it selves

(b). Explain the binary set operation in relational algebra with example.

- Expresses functions from sets to a set.
- Basic Set Operators
    - union, intersection, difference, but no complement. (watch for comparable sets).
- Binary operations
- Result is table(set) with same attributes
- Sets must be compatible!

$R_1(A1,A2,A3)$, $R_2(B1,B2,B3)$

$\text{Domain}(A_i)=\text{Domain}(B_i)$

(c). Distinguish between equijoin and natural join.Equi join

- Equi join is the first type of Inner Join.
- It joins two or more tables where the specified columns are equal.
- In this type of join, you can only use '=' operator in comparing the columns.
- Operators like '>', '<' are not allowed in this type of join.
- Example query: Select * from Employee emp

INNER JOIN Area area on area.EmpId = emp.EmpIdResult:

EmpId	EmpName	AreaId	AreaName	EmpId
3	Peter	4	Canada	3
3	Peter	5	Australia	3
4	Eric	6	England	4

Natural Join:

It is same as equijoin but the difference is that in natural join, the common attribute appears only once

Example query: Select * from Employee emp

NATURAL JOIN area area on area EmpId= emp.EmpId

Result:

EmpId	EmpName	AreaId	AreaName
3	Peter	4	Canada
3	Peter	5	Australia
4	Eric	6	England

OR

VI. (a) Differentiate relational schema and relation with an example.

Relation schema:

- A set of attributes is called a relational schema.
- A relational schema is also known as a table.
- It is the logical definition of table.
- It defines what the name of the table is.

Relational schema:

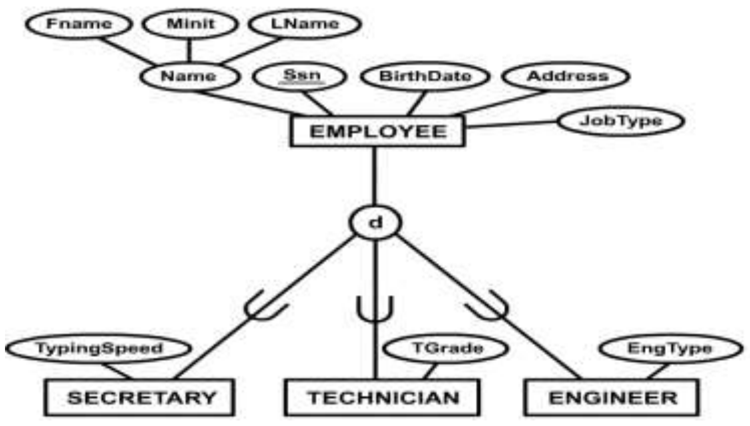
- It is database schema
- A database schema is a collection of relation schemas for a whole database.
- Relational or database schema is a collection of meta-data.

(b). Explain the following with diagram :

i. Specialization

- Is the process of defining a set of subclasses of a superclass
- The set of subclasses is based upon some distinguishing characteristics of the entities in the superclass
- Example: {SECRETARY, ENGINEER, TECHNICIAN} is a specialization of EMPLOYEE based upon *job type*.

May have several specializations of the same superclass



ii. **Generalization**

- The reverse of the specialization process
- Several classes with common features are generalized into a superclass; original classes become its subclasses
- Example: CAR, TRUCK generalized into VEHICLE; both CAR, TRUCK become subclasses of the superclass VEHICLE.
- We can view {CAR, TRUCK} as a specialization of VEHICLE
- Alternatively, we can view VEHICLE as a generalization of CAR and TRUCK

UNIT III

VII. Define normalization. Explain the various normal forms

- This is the process which allows you to winnow out redundant data within your database.
- This involves restructuring the tables to successively meeting higher forms of Normalization.
- A properly normalized database should have the following characteristics
  - Scalar values in each fields
  - Absence of redundancy.
  - Minimal use of null values.

- Minimal loss of information.

- **First Normal Form (1NF)**
  - A table is considered to be in 1NF if all the fields contain
  - only scalar values (as opposed to list of values).
- **Second Normal Form (2NF)**
  - For a table to be in 2NF, there are two requirements
  - The database is in first normal form
  - All **non key** attributes in the table must be functionally dependent on the entire primary key
  - ***Note:*** *Remember that we are dealing with non-key attributes*
- **Third Normal Form (3NF)**
  - This form dictates that all **non-key** attributes of a table must be functionally dependent on a candidate key i.e. there can be no interdependencies among non-key attributes.
  - For a table to be in 3NF, there are two requirements
  - The table should be second normal form
  - No attribute is transitively dependent on the primary key
- **Boyce-Codd Normal Form (BCNF)**
  - BCNF does not allow dependencies between attributes that belong to candidate keys.
  - BCNF is a refinement of the third normal form in which it drops the restriction of a non-key attribute from the 3rd normal form.
  - Third normal form and BCNF are not same if the following conditions are true:
  - The table has two or more candidate keys
  - At least two of the candidate keys are composed of more than one attribute

  - The keys are not disjoint i.e. The composite candidate keys share some attributes

- **Fourth Normal Form (4NF)**
  - Fourth normal form eliminates independent many-to-one relationships between columns.
  - To be in Fourth Normal Form,
  - a relation must first be in Boyce-Codd Normal Form.
  - a given relation may not contain more than one multi-valued attribute.

- **Fifth Normal Form (5NF)**
  - Fifth normal form is satisfied when all tables are broken into as many tables as possible in order to avoid redundancy. Once it is in fifth normal form it cannot be broken into smaller relations without changing the facts or the meaning.

OR

VIII. (a). Explain domain and foreign key constraints.

- A domain has a logical definition: e.g.,"USA_phone_numbers" are the set of 10 digit phone numbers valid in the U.S.
- A domain may have a data-type or a format defined for it. The USA_phone_numbers may have a format: (ddd)-ddd-dddd where each d is a decimal digit. E.g., Dates have various formats such as monthname, date, year or yyyy-mm-dd, or dd mm,yyyy etc.
- An attribute designates the role played by the domain. E.g., the domain Date may be used to define attributes "Invoice-date" and "Payment-date".
- Tuples in the *referencing relation* R_1 have attributes FK (called **foreign key** attributes) that reference the primary key attributes PK of the *referenced relation* R_2. A tuple t_1 in R_1 is said to **reference** a tuple t_2 in R_2 if $t_1[FK] = t_2[PK]$.
- A referential integrity constraint can be displayed in a relational database schema as a directed arc from R_1.FK to R_2.

(b). Discuss join dependencies and 5NF

a) a **join dependency** is a constraint on the set of legal relations over a database scheme. A table T is subject to a join dependency if T can always be recreated by joining multiple tables each having a subset of the attributes of T. If one of the tables in the join has all the attributes of the table T, the join dependency is called trivial.

b) The join dependency plays an important role in the Fifth normal form, also known as *project-join normal form*, because it can be proven that if you decompose a scheme Rin tables $R_1$to R_n, the decomposition will be a lossless-join decomposition if you restrict the legal relations on $R$to a join dependency on Rcalled $*(R_1, R_2, \ldots, R_n)$.

c) Another way to describe a join dependency is to say that the set of relationships in the join dependency is independent of each other.

d) Unlike in the case of functional dependencies, there is no sound and complete axiomatization for join dependencies,[1] though axiomatization exist for more expressive dependency languages such as full typed dependencies.[2] However, implication of join dependencies is decidable.

Fifth normal form is satisfied when all tables are broken into as many tables as possible in order to avoid redundancy. Once it is in fifth normal form it cannot be broken into smaller relations without changing the facts or the meaning.

c) List different transaction states.

- active
- Partially commited
- failed
- commited
- aborted

UNIT IV

IX. (a) There are two tables, 'BOOK' and 'PUBLISHER'. Specify the queries for the following operations :

i. Retrieve all the tuples from both relations which has the same publisher id.

SQL>select * from BOOK groupby publisher id;

ii. Retrieve publisher id, publisher name, address and phone number of publishers publishing novels.

SQL>select publisher id, publisher name, address, phone number from PUBLISHER;

iii. **Retrieve title and price of all books published by 'PHI'.**

SQL>select title, price from PUBLISHER where publisher name='PHI';

iv. **Retrieve the details of books with price if any of the books belonging to 'novel'.**

SQL>select details of books,price from BOOK where Book type='novel';

(b). Explain nested queries with example.

Queries can be nested so that the results of one query can be used in another query via a relational operator or aggregation function. A nested query is known as a **subquery**. While joins and other table operations provide computationally superior (i.e. faster)
An alternative in many cases, the use of subqueries introduces a hierarchy in execution which can be useful or necessary. Since you know how to code SELECT statements, you already know how to code a sub query. It's simply a SELECT statement that's coded within another SQL statement. A sub query can return a single value, a result set that contains a single column (single row sub query), or a result set that contains one or more columns (multiple row sub query).

- Get the product name and company name of the products which have the maximum price.

```
SQL>SELECT product_name, company_name FROM product
        WHERE unit_price = (SELECT MAX(unit_price) FROM product);
```

OR

X. (a). Explain DDL, DML, and DCL commands with suitable examples.

- **Data Definition Language (DDL)**
  - Specification notation for defining the database schema

E.g.

create table *account* (
account-number char(10),
balance integer)

  - DDL compiler generates a set of tables stored in a *data dictionary*
  - Data dictionary contains metadata (i.e., data about data)
  - Database schema

Data *storage and definition* language

Language in which the storage structure and access methods used by the database system are specified

Usually an extension of the data definition language

- **Data Manipulation Language (DML)**
- Language for accessing and manipulating the data organized by the appropriate data model
    - A declarative DML is also known as **query language**
- Two classes of languages
    - Procedural – user specifies what data is required and how to get those data (DML)
    - Nonprocedural – user specifies what data is required without specifying how to get those data (Query language)
- SQL is the most widely used query language

DCL

- DCL is abbreviation of **Data Control Language**. It is used to create roles, permissions, and referential integrity as well it is used to control access to database by securing it.
- GRANT – Gives user's access privileges to database
  REVOKE – Withdraws user's access privileges to database given with the GRANT command

(b). Specify the different data types used in SQL. Write a query to create a table with all the data types. Also write a query to display the structure of the table.

- char(n). Fixed length character string, with user-specified length *n.*
- varchar(n). Variable length character strings, with user-specified maximum length *n.*
- int. Integer (a finite subset of the integers that is machine-dependent).
- smallint. Small integer (a machine-dependent subset of the integer domain type).

- numeric(p,d). Fixed point number, with user-specified precision of *p* digits, with *n* digits to the right of decimal point.
- real, double precision. Floating point and double-precision floating point numbers, with machine-dependent precision.
- float(n). Floating point number, with user-specified precision of at least *n* digits.
- An SQL relation is defined using the **create table** command:

create table *r* (A_1 D_1, A_2 D_2, ..., A_n D_n,
(integrity-constraint$_1$),
...,
(integrity-constraint$_k$))

- *r* is the name of the relation
- each A_i is an attribute name in the schema of relation *r*
- D_i is the data type of attribute A_i
- Example:

create table *branch*
(*branch_name* **char**(15)**,**
branch_city **char(**30),
assets **integer**)